Crossing the Rhine: Moving to CLE 6.0 System Management

Tina L. P. Butler
National Energy Research Scientific Computing Center
Ernest Orlando Lawrence Berkeley National Laboratory
Berkeley, CA USA
Email: tlbutler@lbl.gov

Abstract **With the release of Cray Linux Environment 6.0, Cray has introduced a new paradigm for CLE system configuration and management. This major shift requires significant changes in formatting and practices on the System Management Workstation (SMW). Although Cray has committed to delivering migration tools for legacy systems, they will not be available until CLE 6.0 UP02, scheduled for July 2016 release. In the third quarter of 2016, NERSC will be taking delivery of the second phase of its Cori system, with Intel KNL processors. KNL requires CLE 6.0. In order to support phase 2, Cori will have to be upgraded to CLE 6.0 - the hard way. This paper will chronicle that effort.**

I. Introduction

Cray's latest version of the Cray Linux Environment (CLE) includes a radical change in system management software and procedures. As existing and new Cray sites move to installing CLE 6.0 and SMW 8.0 they will have to reorient their approach to system upgrades and maintenance to succeed with this new software release. The National Energy Research Scientific Computing Center (NERSC) at Lawrence Berkeley National Laboratory is required to be an early adopter of CLE 6.0/SMW 8.0 . NERSC installed the first phase of it's newest Cray supercomputer, Cori, in the middle of 2015; the delivery of the second phase of Cori is imminent. This second phase is based on Intel's KNL processor, which is only supported in CLE 6.0. In order to install Cori Phase 2, NERSC must move Cori Phase 1 to CLE 6.0.

II. Setting

NERSC is the Department of Energy's largest open science computing centers. NERSC has more than 5000 users and more that 700 user-written applications.

This large and diverse user population is currently supported on three Cray systems that are installed at NERSC: Mendel, a Cray CS300; Edison, a Cray XC-30 and Cori, a Cray XC-40 system. Each of the XC systems has one or more large Lustre-based scratch filesystems. Persistent storage for users is provided by the NERSC Global Filesystem (NGF), which comprises 8 GPFS-based filesystem instances that are mounted on all of NERSC's production and testbed systems. On the Cray systems compute node access to the NGF filesystems is provided through the Data Virtualization Service (DVS).

III. CLE 6.0 – Why upgrade now?

Cori Phase 1, delivered to NERSC in 2015, consists of 12 cabinets with Intel Haswell-based compute nodes and 144 Cray DataWarp nodes. The focus of the phase 1 system was very much to support an emerging data-intensive workload. Since the DataWarp software was not yet ready for CLE 6.0 when Cori P1 arrived, the system was brought into production with CLE 5.2 UP04. Cori P2, arriving in the summer of 2016, has 52 cabinets of KNL compute nodes and another 144 DataWarp nodes. In order to install Phase 2 and integrate it with Phase 1, it will be necessary to bring the Phase 1 system up to the CLE 6.0 release.

At the time that Cori Phase 1 was delivered the only CLE 6.0 release available was CLE 6.0 UP00. CLE 6.0 UP00 was a limited availability (LA) release that was only intended for newly installed XC-40 systems. In order to limit system downtime, it was decided that NERSC should install CLE 6.0/SMW 8.0 UP00 on the Cori test system (TDS), Gerty, as a learning and debugging exercise.

IV. Cray Management System Basics

CLE 6.0/SMW 8.0, code named Rhine/Redwood introduces a new style of system management, the Cray Management System (CMS). CMS represents a redesign of the whole CLE/SMW install and upgrade process using more recent and more widely known tools like Ansible and Open Stack. CMS separates the installation and configuration of software images for service, login and compute nodes, allowing prescriptive definition of host configurations and local custom node types. Base images for both internal and external hosts are created using repositories and recipes on the SMW. Host-specific configuration data is created using worksheets and the configurator, and then applied at boot time through a set of ansible plays.

V. NERSC Local Customizations

NERSC has a number of localized customizations in place on Cori under CLE 5.2, but two particular local configurations have to be adapted to work with CMS.. NERSC network configuration is not accomplished through node specialization, but instead uses a series of local scripts and interface and route definition files to set IP addresses and routes at boot time.. The other significant local variation is the use of DVS to serve the GPFS-based NERSC global filesystems (NGF) to compute and service nodes on Cori. NGF center-wide filesystems provide all home directories and persistent storage on all of NERSC's production systems.

VI. Network Configuration

Cori has 32 RSIP nodes, 32 DVS nodes 2 network nodes and 130 LNET nodes. Using the configurator (cfgset) is laborious and error-prone, since the data for each interface has to be entered over multiple queries. Bonded interfaces are not yet supported in the cfgset syntax, so we needed to work around that limitation as well. Local Cray support staff wrote a configuration scraper script that collected information from all the service and login nodes on Cori and Gerty. The scraper data from Gerty was then used to populate the cray_net_worksheet.yaml file with data for all the network interfaces and route information. Interface bonding was accomplished by constructing ifcfg files that were uploaded to nodes using the CMS provided simple_sync service and scripts run by ansible plays at boot time. This was ultimately successful, but efficient network configuration is still a work in progress.

VII. GPFS at NERSC

The NERSC Global Filesystem is actually 8 different GPFS-based filesystem instances that are mounted on all NERSC production systems. Each system has its own GPFS remote clusters that access the filesystems through the central owning clusters. On Cori and Gerty, DVS and elogin nodes mount GPFS natively using the standard GPFS client; the DVS nodes serve the NGF filesystems to the compute partition and selected service nodes.

One of the features of CLE 6.0 is that it is based on SLES 12. Only the two latest releases of GPFS are supported on SLES 12, 4.1.1 and 4.2.0. Going to CLE 6.0 will require a GPFS upgrade as well as an operating system upgrade. Fortunately, the NGF owning clusters are currently being upgraded to GPFS 4.1.1, so they will be able to support the required versions for SLES 12.

VIII. GPFS with Rhine/Redwood

GPFS installations and upgrades need to be maintainable and sustainable. To install GPFS under CLE 6.0, we could do manual workarounds, but that does not lead to a sustainable model. Some of the changes coming with UP01 should help, but further modifications will probably be needed. The GPFS install model is not a natural fit with the CLE 6.0 install philosophy. GPFS requires an initial install of the base RPMS, and then subsequent update installs for any fix levels (PTFs). After the necessary RPMs are installed into the bootable image, a personality layer has to be built on a booted client node; this personality RPM has to be regenerated whenever the kernel changes and reapplied to the bootable image. The GPFS remote cluster configuration must also persist across boots.

IX. Installing GPFS

The first step in the GPFS installation was to create local repositories for GPFS RPMs. I created a repository for both the base release – in this case, 4.1.0 – and update RPMs – 4.1.1.0 and 4.1.1.4. CMS repositories are managed using the **repo** command:

repo create --arch x86-64 --type SLES12 gpfs-4.1-base

Desired RPMs are then added to the repositories. I also created two package collections with the subset of gpfs RPMs needed for the base and update installs. GPFS installation cannot rely on dependencies since the base requires an initial install (rpm –ivh) and the updates require an update install (rpm –Uvh). I discovered that package collection syntax is quite picky , so several iterations were required before all required RPMs were correctly installed in the new image. Validating repos and pkgcolls is not a guarantee that they will work in the anticipated fashion.

Image recipes are JSON files that define the set of rpms needed to create an image. Recipes can contain nested package collections and packages. The set of recipes for an architecture or release are grouped together in a single JSON file. You can see the entire set of existing recipes using the **recipe list** command. I cloned the existing base service node recipe to modify for the GPFS client recipe:

recipe create –clone service_cle_6.0up00_sles_12_x86-64-ari nersc_gpfs_client

The local recipe is written into /etc/opt/cray/imps/image_recipes.d/image_recipes.local.json. I then added the gpfs-base pkgcoll and repo to the cloned recipe. To create the new custom image:

image create –r nersc_gpfs_client nersc_gpfs_client

The new image is written to /var/opt/cray/imps/image_roots. An image root is a directory hierarchy with all the installed RPMs from the recipe. The install can be checked by chroot'ing into the install image and querying the RPM database.

At this point I had to cheat. I copied the gpfs update RPMs to image_root/nersc_gpfs_client/tmp and installed them manually. I now had an image root with GPFS installed, so I tried a test run of building the personality layer or "shim", which consists of a set of kernel modules. I discovered that the base service node recipe does not include several things required for building the shim – make, gcc and the kernel header files.

I found the necessary RPMs in the release repositories, added them to the custom recipe, and built a new image root. Another test run of building the shim failed when the kernel header version.h could not be found. I eventually found version.h and added a symbolic link:

```
ln –s /usr/src/linux-3.12.48-52.27/include/uapi/linux/dvb/version.h /usr/src/linux-3.12.48-52.27/include/linux/version.h
```

The next build attempt was successful. Finally, I was able to create a bootable image:

```
Image export nersc_gpfs_client
```

The bootable image was written to /var/opt/cray/imps/boot_images. I associated the image with a DVS node:

```
Cnode update –I /var/opt/cray/imps/boot_images/nersc_gpfs_client.cpio –n c0-0cs3n2
```

The DVS node was then booted with the new image. The GPFS personality RPM was then built in the proper client environment, and the RPM was harvested from /root/rpmbuild back to the SMW.

```
/usr/lpp/mmfs/bin/mmbuildgpl
/usr/lpp/mmfs/bin/mmbuildgpl –build-package
```

The GPFS client image now has to be rebuilt with the personality RPM that was generated on the DVS server. Every time the kernel changes, or GPFS is upgraded, the personality layer must be rebuilt.

X. What remains to be done?

Now that GPFS has been installed, the DVS nodes have to be configured. In UP00, the DVS configuration worksheet and cfgset support special mount options for DVS client nodes, but not for DVS servers. It is necessary to augment /etc/fstab with the entries required to define the NGF mounts. This is accomplished with simple_sync and an ansible play. The remote cluster also has to be built in order to mount the NGF filesystems. The GPFS cluster configuration is stored in /var/mmfs/gen/mmsdrfs. Non-volatile storage defined for each service node will preserve that file under normal circumstances, but we still need a method for disaster recovery.

In GPFS 4.1.1, a new set of features has been introduced that will allow a backup copy of mmsdrfs to be automatically updated every time a configuration change is made. The mmsdrbackup callback can write cluster configuration changes to a specified copy of mmsdrfs on the boot raid; a companion command, mmsdrrestore will recreate the cluster from the backup copy. This has not been implemented yet on CLE 6.0, but it has been demonstrated on NERSC's GPFS development cluster.

XI. Summary

CLE 6.0/SMW 8.0 represent a large change in system management philosophy and practice. It is very much a work in progress, and we look forward to improvements that will be available shortly in UP01. In particular, supporting NERSC's GPFS center-wide infrastructure is a partially solved problem and has plenty of room for improvements in ongoing work with Rhine/Redwood.

Acknowledgment

This work was supported by the Director, Office of Science, Office of Advance Scientific Compting Research of the U.S. Department of Energy under contract No. DEAC02-05CH11231.

References

[1] "Cray Management System for XC Systems with SMW 8.0/CLE 6.0 (Draft)", Harold Longley, May 2016.

[2] IBM Spectrum Scale Concepts, Planning, and Installation Information, IBM Corporation 2016